REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 27, 1994	Final Report, 9/28/92-5/31/94

4. TITLE AND SUBTITLE

An Introduction to the Principles of Computer Science:
A Reuse -Oriented Philosophy

5. FUNDING NUMBERS

DAAL03-92-G-0412

6. AUTHOR(S)

Dr. Murali Sitaraman (West Virginia University)
Dr. Douglas E. Harms (Muskingum College)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Department of Statistics and Computer Science
PO Box 6330
West Virginia University
Morgantown, WV 26505-6330

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

ARO 30998.4 -MA

11. SUPPLEMENTARY NOTES

The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The long-term goal of our project is to redesign the software development courses in computer science around a philosophy of software reuse, where new components are constructed largely from assembling existing ones. The emphasis is on formal specification, design, and development of "highquality" reusable components. Ada, when augmented with the RESOLVE technology for component-based software development, facilitates introduction this approach. This result has been successfully demonstrated in principle and in practice for the second course in computer science under this project.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Ada, Software reuse, Software Engineering, Undergraduate CS Education	10
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

An Introduction to the Principle of Computer Science — A Reuse-Oriented Philosophy

ARO Proposal Number: 30998-MA; Contract Number: DAAL03-92-G-0412

Principal Investigators: Murali Sitaraman and Douglas E. Harms

Foreword

There is growing consensus that to overcome its crisis, the software industry must make serious strides in adopting the successful practices of more mature engineering disciplines. This would include, for example, injecting formal methods into the software-development process and whole-heartedly embracing the idea of building software from off-the-shelf reusable components.

Towards addressing the software's chronic crisis, over the past several years, the reusable software research groups at The Ohio State University and West Virginia University have been investigating the software development philosophy whereby a software system is properly conceptualized and constructed as an assemblage of reusable components. As a result of these research efforts, a conceptually robust and sound discipline — referred to as RESOLVE — for the design, specification, implementation, verification, and application of reusable software components — has evolved [SIGSOFT 94].

The long-term goal of the our overall research program is the integration of component-based RESOLVE software technology into the entire software development sequence of the undergraduate curriculum, and evaluate the influences of the appproach. The current project showed the feasibility of the successfully demonstrated the feasibility of this view by redesigning the second course in computer science, without eliminating the principles traditionally taught in the course.

2. Table of Contents

3. List of Appendices

Appendix A — List of Deliverables mailed earlier as of July 1993

Appendix B — An Outline for the First Course in Computer Science

4. Body of the FIinal Report: DAAL03-92-G-0412

A. Statement of the Problem Studied

The long-term goal of our project is to redesign the fundamental courses in computer science around a philosophy of software reuse and evaluate the influences of this approach. The emphasis here is on the specification, design, and development of "high quality" reusable software components, and construction of new software by assembling existing reusable components rather than starting from scratch.

The reuse-centered approach is aimed at remedying the following fundamental problems in undergraduate education in computer science:

1. Absence of a context, and hence motivation, for learning fundamental principles of computer science such as abstraction, early in the curriculum,
2. Absence of a context for emphasizing and teaching that specification and design issues are central to problem-solving activity early in the curriculum
3. Relatively late exposure to principles of software engineering resulting in relatively inexperienced graduates in applying these principles,
4. Generation of students underprepared to function effectively in a component-based software industry, and
5. Minimal exposure to software engineering principles (without any good context) for non-computer science majors.

The focus of the present project was to *redesign the second course in computer science* folowing a reuse-centered approach, without eliminating the principles traditionally taught in the course. The project has resulted in materials for teaching this course in *Ada.*

B. Summary of the Most Important Results

1. Focus on reuse provides an excellent context for presenting important computer science principles.

The idea that a software component will be reused elsewhere permits the students to see readily the importance of key software engineering principles. The realization that the developer of a component and its prospective client are likely to be different people leads the students to new thinking. In particular, the importance and relevance of the following principles become clear to students early in their curriculum.

- Separation of the specification and implementation details of a component
- Unambigous and abstract expression of a specification
- Design
- Certification of correctness
- Efficient implementations
- Maintenance

2. Focus on reuse permits introduction to principles of specification and design early in the curriculum.

In the reuse-based course, the laboratory instructor and the student form a team. In some projects, the student uses a reusable component (that has already been implemented by the instructor) to solve a problem. In others, a student is the developer of a component. This permits students see the importance of the principles of design and specification early in the

curriculum.

3. Freshmen students can understand and use formal specifications, if an "understandable" specification language such as RESOLVE is used [SIGSOFT 94]*; They are also able to master the RESOLVE design discipline for Ada.*

4. Ada, augmented with formal specifications, is an excellent vehicle for teaching software engineering and software reuse principles in computer science curriculum.

5. Non-computer science majors get an important insight into principles of software reuse and software engineering.

6. The materials are appropriate for a variety of institutions

7. The principles contained in the materials are applicable for multiple programming languages, even though the materials themselves use Ada

C. List of Publications

Only direct results are reported here.

1. Sitaraman, M., *Principles of Computer Science — CS 16 Course Notes*, Spring 1994, 150 pages approx. (Enclosed)

2. Sitaraman, M. and Gray, J., Software Reuse: A Context for Introducing SE Principles in a Traditional CS Second Course," *Proceedings of TriAda 1993*, Seattle, WA, September 1993, to appear.

3. Gray, J., "Teaching the Second Course in Computer Science in a Reuse-Based Setting: A Sequence of Lab Assignments in Ada," *Proceedings of the Eleventh National Conference on Ada Technology*, Wiiliamsburg, VA, 38-45, March 1993.

D. List of Participating Personnel

Dr. Murali Sitaraman (West Virginia University)
Dr. Douglas E. Harms (Musingum College)
Mr. Jeff Gray (MS Student, West Virginia University)

Gray, J., *The Role of Reuse in Introducing Software Engineering Principles in a Computer Science Second Course*, Masters Project Report, Dept. of Statistics and Computer Science, WVU, Morganotown, WV, May 1993.

5. Report of Inventions

M. Sitaraman, *Principles of Computer Science — CS 16 Course Notes*, Barnes and Noble, Morgantown, West Virginia, 1994.

6. References in the Final Report

[SIGSOFT 94] Special Feature: Component-Based Software Using RESOLVE, *ACM SIGSOFT Software Engineering Notes*, vol. 19, no. 4, eds. M. Sitaraman and B. W. Weide, October 1994, 21-67.

7. Appendix A — List of Deliverables mailed earlier as of July 1993

1. Sitaraman, M., *Principles of Computer Science — CS 16 Course Notes*, 1993, 170 pages approx.

   These notes have been developed as explained in the proposal. They include 12 organized chapters. They are now being used at the West Virginia University. Notes have also been sent to Muskingum College, Indiana University Southeast, and The Ohio State University for possible adaptation.

2. Assignments/Exercises

   These can be found at the end of the 12 chapters.

3. Exams

   A set of 3 exams is enclosed.

4. Overhead materials

   One re-producible set of 59 overheads is enclosed.

5. Lab Sequences for the Second Course in Computer Science

   2 different sets are enclosed. Also see the information on publications.

6. Results of experimentation

   The second course was taught following the reuse-centered approach at West Virginia University by Sitaraman in Spring 1993 and is currently being taught again in Summer 93 by instructor Winsome Mundy.

7. Student feedback

   From the confidential evaluations, we note that the student feedback has been positive for the offering in Spring 93. 8 students evaluated the course as good or excellent and 4 found the course fair or satisfactory.

   Most students found the course notes "very valuable," though they are still evolving. Some of them actually suggested that the notes be made into a textbook and were better than the textbook they used in their first computer science course. Efforts on this front are underway. Some students found the labs harder than the others. A look at the final grades suggests that the performance of the students by way of grades was about average. The significant influences of the current course, of course, are expected to be in their later courses. More information can be found in the publications.

8. Other textbooks and comparison

   A comparison of notes with other popular textbooks and approaches can be found in Gray's Masters project report.

7. Appendix B — An Outline for the First Course in Computer Science

1. Introduction
    - Why write computer programs?
    - Reusing recurring program pieces
    - The engineering metaphor and software construction

2. Formal Specification
    - Modeling
    - The Integer Facility
        - Procedures and Procedure Calls
        - Packages: with and use Clauses
        - Variables
        - Initialize and Finalize Operations
        - Using the Integer Facility
    - The Prompt Facility
    - Pre-conditions and Post-conditions
    - Parameter/Variable States
    - Logic

3. Application of components
    - Introduction to secondary operations and layered components
    - Applications - The Hardware Store Example
    - Formal Reasoning of programs using reusable components

4. Decisions
    - The Boolean Facility
    - The if Statement
    - Reasoning about decisions

5. Iteration
    - The indefinite loop statement
    - The definite loop statement
    - Reasoning about loops

6. Character Strings
    - Specification of the Character String Facility
    - Applications

8. Generic Structures and Arrays
    - Specification of Array_Template
    - Application - Introduction to searching and sorting
    - Reasoning of Programs Using Arrays

9. Records

10. Ada Specific Features
    - Built-in Integer, Boolean, Character, String, Array, Record
    - Floating points
    - Parameter modes - in, out, in out
    - Functions
    - elsif portion of if statement
    - Unconstrained arrays; Discriminant Records